Aprendendo python - Unidade 3

Classes e Métodos em python

ABSTRAÇÃO - CLASSES E OBJETOS

objetos são os componentes de um programa OO. Um programa que usa a tecnologia OO é basicamente uma coleção de objetos. Uma classe é um modelo para um objeto. Podemos considerar uma classe uma forma de organizar os dados (de um objeto) e seus comportamentos (PSF, 2020a). Vamos pensar na construção de uma casa: antes do "objeto casa" existir, um arquiteto fez a planta, determinando tudo que deveria fazer parte daquele objeto. Portanto, a classe é o modelo e o objeto é uma instância. Entende-se por instância a existência física, em memória, do objeto.

ATRIBUTOS

Os dados armazenados em um objeto representam o estado do objeto. Na terminologia de programação OO, esses dados são chamados de atributos. Os atributos contêm as informações que diferenciam os vários objetos – os funcionários, neste caso.

MÉTODOS

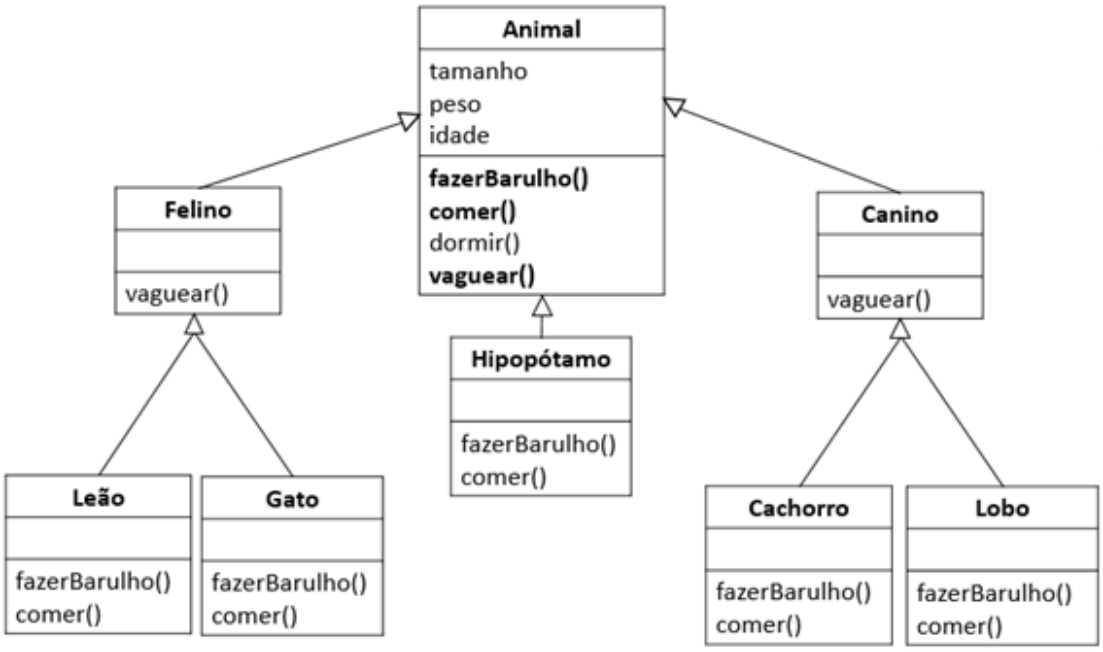
O comportamento de um objeto representa o que este pode fazer. Nas linguagens procedurais, o comportamento é definido por procedimentos, funções e sub-rotinas. Na terminologia de programação OO, esses comportamentos estão contidos nos métodos, aos quais você envia uma mensagem para invocá-los.

ENCAPSULAMENTO

O ato de combinar os atributos e métodos na mesma entidade é, na linguagem OO, chamado de encapsulamento (Weisfeld, 2013), termo que também aparece na prática de tornar atributos privados, quando estes são encapsulados em métodos para guardar e acessar seus valores.

HERANÇA

Por meio desse mecanismo, é possível fazer o reúso de código, criando soluções mais organizadas. A herança permite que uma classe herde os atributos e métodos de outra classe. Observe, na Figura 3.2, que as classes funcionário e cliente herdam os atributos da classe pessoa. A classe pessoa pode ser chamada de classe-pai, classe-base, superclasse, ancestral; por sua vez, as classes derivadas são as classes-filhas, subclasses.



POLIFORMISMO

Polimorfismo é uma palavra grega que, literalmente, significa muitas formas. Embora o polimorfismo esteja fortemente associado à herança, é frequentemente citado separadamente como uma das vantagens mais poderosas das tecnologias orientadas a objetos (WEISFELD, 2013). Quando uma mensagem é enviada para um objeto, este deve ter um método definido para responder a essa mensagem. Em uma hierarquia de herança, todas as subclasses herdam as interfaces de sua superclasse. No entanto, como toda subclasse é uma entidade separada, cada uma delas pode exigir uma resposta separada para a mesma mensagem.

In [ ]:
```
class PrimeiraClasse:
    def imprimir_mensagem(self, nome):
        print(f"Olá {nome}, seja bem vindo!")

pessoa1 = PrimeiraClasse()
pessoa1.imprimir_mensagem('Ryan')
```

```
Olá Ryan, seja bem vindo!
```

In [ ]:
```
class Calculadora:

    def somar(self, n1, n2):
        return n1 + n2

    def subtrair(self, n1, n2):
        return n1 - n2

    def multiplicar(self, n1, n2):
        return n1 * n2

    def dividir(self, n1, n2):
        return n1 / n2

    def dividir_resto(self, n1, n2):
        return n1 % n2
```

```
calc = Calculadora()

print('Soma:', calc.somar(4, 3))
print('Subtração:', calc.subtrair(13, 7))
print('Multiplicação:', calc.multiplicar(2, 4))
print('Divisão:', calc.dividir(16, 5))
print('Resto da divisão:', calc.dividir_resto(7, 3))
```

```
Soma: 7
Subtração: 6
Multiplicação: 8
Divisão: 3.2
Resto da divisão: 1
```

CONSTRUTOR DA CLASSE **INIT**()

Até o momento criamos classes com métodos, os quais utilizam variáveis locais. E os atributos das classes? Nesta seção, vamos aprender a criar e utilizar atributos de instância, também chamadas de variáveis de instâncias. Esse tipo de atributo é capaz de receber um valor diferente para cada objeto. Um atributo de instância é uma variável precedida com o parâmetro self, ou seja, a sintaxe para criar e utilizar é self.nome_atributo.

In [ ]:
```
class Televisao:
    def __init__(self):
        self.volume = 10

    def diminuir_volume(self, v):
        self.volume = self.volume - v

    def aumentar_volume(self, v):
        if self.volume < 10:
            self.volume = self.volume + v
        else:
            print("volume ja está no máximo!")

tv = Televisao()
print("Volume ao ligar a tv = ", tv.volume)
tv.aumentar_volume(3)
print("Volume atual ", tv.volume)
tv.diminuir_volume(5)
print("Volume atual ", tv.volume)
```

```
Volume ao ligar a tv =  10
volume ja está no máximo!
Volume atual  10
Volume atual  5
```

VARIÁVEIS E MÉTODOS PRIVADOS

Em linguagens de programação OO, como Java e C#, as classes, os atributos e os métodos são acompanhados de modificadores de acesso, que podem ser: public, private e protected. Em Python, não existem modificadores de acesso e todos os recursos são públicos. Para simbolizar que um atributo ou método é privado, por convenção, usa-se um sublinhado "_" antes do nome; por exemplo, _cpf, _calcular_desconto()

Conceitualmente, dado que um atributo é privado, ele só pode ser acessado por membros da própria classe. Portanto, ao declarar um atributo privado, precisamos de métodos que acessem e recuperam os valores ali guardados. Em Python, além de métodos para este fim, um atributo privado pode ser acessado por decorators.

In [ ]:
```
class ContaCorrente:
    def __init__(self):
        self.saldo = None

    def depositar(self, valor):
        self._saldo += valor

    def sacar(self, valor):
        self._saldo -= valor

    def consultarSaldo(self):
        return self._saldo
```

In [ ]:
```
class Pessoa:
    def __init__(self):
        self.cpf = None
        self.nome = None
        self.endereco = None

class Funcionario(Pessoa):
    def __init__(self):
        self.matricula = None
        self.salario = None
        self.senha = None

    def bater_ponto(self):

        pass

    def fazer_login(self):

        pass

class Cliente(Pessoa):
    def __init__(self):
        self.codigo = None
        self.data_cadastro = None

    def realizar_cadastro():

        pass

    def fazer_compra(self):

        pass

    def pagar_conta(self):

        pass

#Programa
```

Bibliotecas e modulos em python

MÓDULOS E BIBLIOTECAS EM PYTHON

Uma opção para organizar o código é implementar funções, contexto em que cada bloco passa a ser responsável por uma determinada funcionalidade. Outra forma é utilizar a orientação a objetos e criar classes que encapsulam as características e os comportamentos de um determinado objeto. Conseguimos utilizar ambas as técnicas para melhorar o código, mas, ainda assim, estamos falando de toda a solução agrupada em um arquivo Python (.py).

Para utilizar um módulo é preciso importá-lo para o arquivo. Essa importação pode ser feita de maneiras distintas:

- import moduloXXText
- import moduloXX as apelido
- from moduloXX import itemA, itemB

```
import math

math.sqrt(25)
math.log2(1024)
math.cos(45)

import math as m

m.sqrt(25)
m.log2(1024)
m.cos(45)

from math import sqrt, log2, cos

sqrt(25)
log2(1024)
cos(45)
```

CLASSIFICAÇÃO DOS MÓDULOS (BIBLIOTECAS)

Podemos classificar os módulos (bibliotecas) em três categorias, cada uma das quais vamos estudar:

- Módulos built-in: embutidos no interpretador.
- Módulos de terceiros: criados por terceiros e disponibilizados via PyPI.
- Módulos próprios: criados pelo desenvolvedor.

MÓDULOS BUILT-IN

Ao instalar o interpretador Python, também é feita a instalação de uma biblioteca de módulos, que pode variar de um sistema operacional para outro.

MÓDULO RANDOM

Random é um módulo built-in usado para criar número aleatórios. Vamos explorar as funções:

- random.randint(a, b): retorna um valor inteiro aleatório, de modo que esse número esteja entre a, b.
- random.choice(seq): extrai um valor de forma aleatória de uma certa sequência.
- random.sample(population, k): retorna uma lista com k elementos, extraídos da população.

In [ ]:
```
import random

print(random.randint(0, 100))
print(random.choice([1, 10, -1, 100]))
print(random.sample(range(100000), k=12))
```

```
31
10
[65251, 52388, 16946, 48344, 24898, 44606, 53650, 78937, 10640, 31081, 9537, 27
853]
```

MÓDULO OS

OS é um módulo built-in usado para executar comandos no sistema operacional. Vamos explorar as funções:

- os.getcwd(): retorna uma string com o caminho do diretório de trabalho.
- os.listdir(path='.'): retorna uma lista com todas as entradas de um diretório. Se não for especificado um caminho, então a busca é realizada em outro diretório de trabalho.
- os.cpu_count(): retorna um inteiro com o número de CPUs do sistema.
- os.getlogin(): retorna o nome do usário logado.
- os.getenv(key): retorna uma string com o conteúdo de uma variável de ambiente especificada na key.
- os.getpid(): retorna o id do processo atual.

In [ ]:
```
import os

os.getcwd()
os.listdir()
os.cpu_count()
os.getlogin()
os.getenv(key='path')
os.getpid()
```

Out[ ]:
```
9224
```

MÓDULO RE

O módulo re (regular expression) fornece funções para busca de padrões em um texto. Uma expressão regular especifica um conjunto de strings que corresponde a ela. As funções neste módulo permitem verificar se uma determinada string corresponde a uma determinada expressão regular. Essa técnica de programação é utilizada em diversas linguagens de programação, pois a construção de re depende do conhecimento de padrões. Vamos explorar as funções:

- re.search(pattern, string, flags=0): varre a string procurando o primeiro local onde o padrão de expressão regular produz uma correspondência e o retorna. Retorna None se nenhuma correspondência é achada.
- re.match(pattern, string, flags=0): procura por um padrão no começo da string. Retorna None se a sequência não corresponder ao padrão.
- re.split(pattern, string, maxsplit=0, flags=0): divide uma string pelas ocorrências do padrão.

Para entendermos o funcionamento da expressão regular, vamos considerar um cenário onde temos um nome de arquivo com a data: meuArquivo_20-01-2020.py. Nosso objetivo é guardar a parte textual do nome em uma variável para a usarmos posteriormente. Vamos utilizar os três métodos para fazer essa separação. O search() faz a procura em toda string, o match() faz a procura somente no começo (razão pela qual, portanto, também encontrará neste caso) e o split() faz a transformação em uma lista. Como queremos somente a parte textual, pegamos a posição 0 da lista.

In [ ]:
```
import re

string = 'meuArquivo_20-01-2020.py'
padrao = "[a-zA-Z]*"

texto1 = re.search(padrao, string).group()
texto2 = re.match(padrao, string).group()
texto3 = re.split("_", string)[0]

print(texto1)
print(texto2)
print(texto3)
```

```
meuArquivo
meuArquivo
meuArquivo
```

Na linha 4, da entrada 6, construímos uma expressão regular para buscar por sequências de letras maiúsculas e minúsculas [a-zA-Z], que pode variar de tamanho 0 até N (*). Nas linhas 6 e 7 usamos esse padrão para fazer a procura na string. Ambas as funções conseguiram encontrar; e, então, usamos a função group() da re para capturar o resultado. Na linha 8, usamos o padrão "_" como a marcação de onde cortar a string, o que resulta em uma lista com dois valores – como o texto é a primeira parte, capturamos essa posição com o [0].

MÓDULO DATETIME

Trabalhar com datas é um desafio nas mais diversas linguagens de programação. Em Python há um módulo built-in capaz de lidar com datas e horas. O módulo datetime fornece classes para manipular datas e horas. Uma vez que esse módulo possui classes, então a sintaxe para acessar os métodos deve ser algo similar a: modulo.classe.metodo(). Dada a diversa quantidade de possibilidades de se trabalhar com esse módulo, vamos ver um pouco das classes datetime e timedelta.

In [ ]:
```
import datetime as dt

# Operações com data e hora
hoje = dt.datetime.today()
ontem = hoje - dt.timedelta(days=1)
uma_semana_atras = hoje - dt.timedelta(weeks=1)

agora = dt.datetime.now()
duas_horas_atras = agora - dt.timedelta(hours=2)

# Formatação
hoje_formatado = dt.datetime.strftime(hoje, "%d-%m-%Y") #https://docs.python.org
ontem_formatado = dt.datetime.strftime(ontem, "%d de %B de %Y")

# Converção de string para data
data_string = '11/06/2019 15:30'
data_dt = dt.datetime.strptime(data_string, "%d/%m/%Y %H:%M")
```

Na entrada 7, usamos algumas funcionalidades disponíveis no módulo datetime. Repare que fizemos a importação com a utilização do apelido de dt, prática essa que é comum para nomes grandes.

Linha 4: usamos o método today() da classe datetime para capturar a data e a hora do sistema.

Linha 5: usamos a classe timedelta para subtrair 1 dia de uma data específica.

Linha 6: usamos a classe timedelta para subtrair 1 semana de uma data específica.

Linha 8: usamos o método now() da classe datetime para captura a data e hora do sistema.

Linha 9: usamos a classe timedelta para subtrair 2 horas de uma data específica.

Linhas 12 e 13: usamos o método strftime() da classe datetime para formatar a aparência de uma data específica. [Acesse o endereço https://bit.ly/2E33mzR para verificar as possibilidades de formatação.]

Linha 17: usamos o método strptime() da classe datetime, para converter uma string em um objeto do tipo datetime. Essa transformação é interessante, pois habilita as operações que vimos.

MÓDULOS DE TERCEIROS

Para utilizar uma biblioteca do repositório PyPI, é preciso instalá-la. Para isso, abra um terminal no sistema operacional e digite: pip install biblioteca [biblioteca é o nome do

pacote que deseja instalar. Por exemplo: pip install numpy.]

No dia a dia, exitem bibliotecas que têm sido amplamente utilizadas, como as para tratamento e visualização de dados, para implementações de inteligência artificial (deep learning e machine learning), para tratamento de imagens, para conexão com banco de dados, dentre outras. Veja algumas a seguir:

Bibliotecas para tratamento de imagens

Pillow: esta biblioteca oferece amplo suporte aos formatos de arquivo, uma representação interna eficiente e recursos de processamento de imagem bastante poderosos.

OpenCV Python: é uma biblioteca de código aberto licenciada por BSD que inclui várias centenas de algoritmos de visão computacional.

Luminoth: é um kit de ferramentas de código aberto para visão computacional. Atualmente, atua com a detecção de objetos, mas a ideia é expandi-la.

Mahotas: é uma biblioteca de algoritmos rápidos de visão computacional (todos implementados em C++ para ganhar velocidade) que opera com matrizes NumPy.

Bibliotecas para visualização de dados

Matplotlib: é uma biblioteca abrangente para criar visualizações estáticas, animadas e interativas em Python.

Bokeh: é uma biblioteca de visualização interativa para navegadores modernos. Oferece interatividade de alto desempenho em conjuntos de dados grandes ou de streaming.

Seaborn: é uma biblioteca para criar gráficos estatísticos em Python.

Altair: é uma biblioteca declarativa de visualização estatística para Python.

Bibliotecas para tratamento de dados

Pandas: é um pacote Python que fornece estruturas de dados rápidas, flexíveis e expressivas, projetadas para facilitar o trabalho com dados estruturados (em forma de tabela).

NumPy: além de seus óbvios usos científicos, a NumPy também pode ser usada como um eficiente recipiente multidimensional de dados genéricos.

Pyspark: Spark é um sistema de computação em cluster rápido e geral para Big Data.

Pingouin: é um pacote estatístico Python baseado em Pandas.

Bibliotecas para tratamento de textos

Punctuation: esta é uma biblioteca Python que removerá toda a pontuação em uma string.

NLTK: o Natural Language Toolkit é um pacote Python para processamento de linguagem natural.

FlashText: este módulo pode ser usado para substituir palavras-chave em frases ou extraí-las.

TextBlob: é uma biblioteca Python para processamento de dados textuais.

Internet, rede e cloud

Requests: permite que você envie solicitações HTTP/1.1 com extrema facilidade. Não há necessidade de adicionar manualmente queries de consulta aos seus URLs ou de codificar os dados PUT e POST: basta usar o método JSON.

BeautifulSoup: é uma biblioteca que facilita a captura de informações de páginas da web.

Paramiko: é uma biblioteca para fazer conexões SSH2 (cliente ou servidor). A ênfase está no uso do SSH2 como uma alternativa ao SSL para fazer conexões seguras entre scripts Python.

s3fs: é uma interface de arquivos Python para S3 (Amazon Simple Storage Service).

Bibliotecas para acesso a bancos de dados

mysql-connector-python: permite que programas em Python acessem bancos de dados MySQL.

cx-Oracle: permite que programas em Python acessem bancos de dados Oracle.

psycopg2: permite que programas em Python acessem bancos de dados PostgreSQL.

SQLAlchemy: fornece um conjunto completo de padrões de persistência, projetados para acesso eficiente e de alto desempenho a diversos banco de dados, adaptado para uma linguagem de domínio simples e Python.

Deep learning - Machine learning

Keras: é uma biblioteca de rede neural profunda de código aberto.

TensorFlow: é uma plataforma de código aberto de ponta a ponta para aprendizado de máquina, desenvolvido originalmente pela Google.

PyTorch: é um pacote Python que fornece dois recursos de alto nível: i) computação de tensor (como NumPy) com forte aceleração de GPU; e ii) redes neurais profundas.

Scikit Learn: módulo Python para aprendizado de máquina construído sobre o SciPy (SciPy é um software de código aberto para matemática, ciências e engenharia).

Biblioteca para jogos - PyGame

PyGame: é uma biblioteca para a construção de aplicações gráficas e aplicação multimídia, utilizada para desenvolver jogos.

BIBLIOTECA REQUESTS

A biblioteca requests habilita funcionalidades do procotolo HTTP, como o get e o post. Dentre seus métodos, o get() é o responsável por capturar informação da internet. A documentação sobre ela está disponível no endereço https://requests.readthedocs.io/pt_BR/latest/. Essa biblioteca foi construída com o intuito de substituir o módulo urllib2, que demanda muito trabalho para obter os resultados. O método get() permite que você informe a URL de que deseja obter informação. Sua sintaxe é: requests.get('https://XXXXXXX'). Para outros parâmetros dessa função, como autenticação, cabeçalhos, etc., consulte a documentação.

In [ ]:
```
import requests

info = requests.get('https://api.github.com/events')
info.headers
```

Out[ ]:
```
{'Server': 'GitHub.com', 'Date': 'Tue, 14 Mar 2023 22:39:48 GMT', 'Content-Typ
e': 'application/json; charset=utf-8', 'Cache-Control': 'public, max-age=60, s-
maxage=60', 'Vary': 'Accept, Accept-Encoding, Accept, X-Requested-With', 'ETa
g': 'W/"5572562ea23fe88a170e00bf19fecfc91501fe957d50117e5d1aaa4184baa1a1"', 'La
st-Modified': 'Tue, 14 Mar 2023 22:34:48 GMT', 'X-Poll-Interval': '60', 'X-GitH
ub-Media-Type': 'github.v3; format=json', 'Link': '<https://api.github.com/even
ts?page=2>; rel="next", <https://api.github.com/events?page=10>; rel="last"',
'x-github-api-version-selected': '2022-11-28', 'Access-Control-Expose-Headers':
'ETag, Link, Location, Retry-After, X-GitHub-OTP, X-RateLimit-Limit, X-RateLimi
t-Remaining, X-RateLimit-Used, X-RateLimit-Resource, X-RateLimit-Reset, X-OAuth
-Scopes, X-Accepted-OAuth-Scopes, X-Poll-Interval, X-GitHub-Media-Type, X-GitHu
b-SSO, X-GitHub-Request-Id, Deprecation, Sunset', 'Access-Control-Allow-Origi
n': '*', 'Strict-Transport-Security': 'max-age=31536000; includeSubdomains; pre
load', 'X-Frame-Options': 'deny', 'X-Content-Type-Options': 'nosniff', 'X-XSS-P
rotection': '0', 'Referrer-Policy': 'origin-when-cross-origin, strict-origin-wh
en-cross-origin', 'Content-Security-Policy': "default-src 'none'", 'Content-Enc
oding': 'gzip', 'X-RateLimit-Limit': '60', 'X-RateLimit-Remaining': '59', 'X-Ra
teLimit-Reset': '1678837188', 'X-RateLimit-Resource': 'core', 'X-RateLimit-Use
d': '1', 'Accept-Ranges': 'bytes', 'Transfer-Encoding': 'chunked', 'X-GitHub-Re
quest-Id': '101E:0D3C:307A3D:36A5A8:6410F7B4'}
```

In [ ]:
```
print(info.headers['date']) # Data de extração
print(info.headers['server']) # Servidor de origem
print(info.encoding) # Encoding do texto
print(info.headers['last-modified']) # Data da última modificação da informação
```

```
Tue, 14 Mar 2023 22:39:48 GMT
GitHub.com
utf-8
Tue, 14 Mar 2023 22:34:48 GMT
```

In [ ]:
```
texto_str = info.text
print(type(texto_str))
texto_str[:100] # exibe somente os 100 primeiros caracteres
```

```
<class 'str'>
```

Out[ ]:
```
'[{"id":"27723106677","type":"CreateEvent","actor":{"id":19557880,"login":"bego
naguereca","display_lo'
```

In [ ]:
```
texto_json = info.json()
print(type(texto_json))
```

```
texto_json[0]
```

```
<class 'list'>
```

Out[ ]:
```
{'id': '27723106677',
 'type': 'CreateEvent',
 'actor': {'id': 19557880,
  'login': 'begonaguereca',
  'display_login': 'begonaguereca',
  'gravatar_id': '',
  'url': 'https://api.github.com/users/begonaguereca',
  'avatar_url': 'https://avatars.githubusercontent.com/u/19557880?'},
 'repo': {'id': 371111447,
  'name': 'valet-testing-integration/integration-tests-target',
  'url': 'https://api.github.com/repos/valet-testing-integration/integration-te
sts-target'},
 'payload': {'ref': 'convert-integration-tests-designer-freestyle-elephant-to-a
ctions-20230314-223446',
  'ref_type': 'branch',
  'master_branch': 'main',
  'description': None,
  'pusher_type': 'user'},
 'public': True,
 'created_at': '2023-03-14T22:34:48Z',
 'org': {'id': 84097682,
  'login': 'valet-testing-integration',
  'gravatar_id': '',
  'url': 'https://api.github.com/orgs/valet-testing-integration',
  'avatar_url': 'https://avatars.githubusercontent.com/u/84097682?'}}
```

In [ ]:
```
# primeiro passo extrair as informações com o request utilizando o método json()

import requests
import datetime as dt

jogos = requests.get('http://worldcup.sfg.io/matches').json()
print(type(jogos))

# segundo passo: percorrer cada dicionário da lista (ou seja, cada jogo) extrain

info_relatorio = []
file = open('relatorio_jogos.txt', "w") # cria um arquivo txt na pasta em que es

for jogo in jogos:
    data = jogo['datetime'] # extrai a data
    data = dt.datetime.strptime(data, "%Y-%m-%dT%H:%M:%SZ") # converte de string
    data = data.strftime("%d/%m/%Y") # formata

    nome_time1 = jogo['home_team_country']
    nome_time2 = jogo['away_team_country']

    gols_time1 = jogo['home_team']['goals']
    gols_time2 = jogo['away_team']['goals']

    linha = f"({data}) - {nome_time1} x {nome_time2} = {gols_time1} a {gols_time
    file.write(linha + '\n') # escreve a linha no arquivo txt
    info_relatorio.append(linha)

file.close() # é preciso fechar o arquivo
info_relatorio[:5]
```

```
---------------------------------------------------------------------------
JSONDecodeError                           Traceback (most recent call last)
File c:\Users\ryanj\AppData\Local\Programs\Python\Python39\lib\site-packages\requests\models.py:971, in Response.json(self, **kwargs)
    970 try:
--> 971     return complexjson.loads(self.text, **kwargs)
    972 except JSONDecodeError as e:
    973     # Catch JSON-related errors and raise as requests.JSONDecodeError
    974     # This aliases json.JSONDecodeError and simplejson.JSONDecodeError

File c:\Users\ryanj\AppData\Local\Programs\Python\Python39\lib\json\__init__.py:346, in loads(s, cls, object_hook, parse_float, parse_int, parse_constant, object_pairs_hook, **kw)
    343 if (cls is None and object_hook is None and
    344         parse_int is None and parse_float is None and
    345         parse_constant is None and object_pairs_hook is None and not kw):
--> 346     return _default_decoder.decode(s)
    347 if cls is None:

File c:\Users\ryanj\AppData\Local\Programs\Python\Python39\lib\json\decoder.py:337, in JSONDecoder.decode(self, s, _w)
    333 """Return the Python representation of ``s`` (a ``str`` instance
    334 containing a JSON document).
    335
    336 """
--> 337 obj, end = self.raw_decode(s, idx=_w(s, 0).end())
    338 end = _w(s, end).end()

File c:\Users\ryanj\AppData\Local\Programs\Python\Python39\lib\json\decoder.py:355, in JSONDecoder.raw_decode(self, s, idx)
    354 except StopIteration as err:
--> 355     raise JSONDecodeError("Expecting value", s, err.value) from None
    356 return obj, end

JSONDecodeError: Expecting value: line 1 column 1 (char 0)

During handling of the above exception, another exception occurred:

JSONDecodeError                           Traceback (most recent call last)
Cell In[21], line 6
      3 import requests
      4 import datetime as dt
----> 6 jogos = requests.get('http://worldcup.sfg.io/matches').json()
      7 print(type(jogos))
      9 # segundo passo: percorrer cada dicionário da lista (ou seja, cada jogo) extraindo as informações

File c:\Users\ryanj\AppData\Local\Programs\Python\Python39\lib\site-packages\requests\models.py:975, in Response.json(self, **kwargs)
    971     return complexjson.loads(self.text, **kwargs)
    972 except JSONDecodeError as e:
    973     # Catch JSON-related errors and raise as requests.JSONDecodeError
    974     # This aliases json.JSONDecodeError and simplejson.JSONDecodeError
--> 975     raise RequestsJSONDecodeError(e.msg, e.doc, e.pos)

JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

MATPLOTLIB

Matplotlib é uma biblioteca com funcionalidades para criar gráficos, cuja documentação está disponível no endereço https://matplotlib.org/. É composta por uma série de exemplos. Vamos utilizar a interface Pyplot para criar um gráfico simples baseado nas informações que salvamos sobre os jogos da Copa do Mundo de Futebol Feminino de 2019.

In [ ]:
```
# ler o arquivo salvo
file = open('relatorio_jogos.txt', 'r')
print('file = ', file, '\n')
info_relatorio = file.readlines()
file.close()

print("linha 1 = ", info_relatorio[0])

# Extrair somente a parta 'dd/mm' da linha
datas = [linha[1:6] for linha in info_relatorio]
print(sorted(datas))

datas_count = [(data, datas.count(data)) for data in set(datas)]
print(datas_count)

%matplotlib inline
import matplotlib.pyplot as plt

eixo_x = datas_count.keys()
eixo_y = datas_count.values()

plt.figure(figsize=(15, 5))
plt.xlabel('Dia do mês')
plt.ylabel('Quantidade de jogos')
plt.xticks(rotation=45)

plt.bar(eixo_x, eixo_y)

plt.show()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[20], line 2
      1 # ler o arquivo salvo
----> 2 file = open('relatorio_jogos.txt', 'r')
      3 print('file = ', file, '\n')
      4 info_relatorio = file.readlines()

File ~\AppData\Roaming\Python\Python39\site-packages\IPython\core\interactivesh
ell.py:282, in _modified_open(file, *args, **kwargs)
    275 if file in {0, 1, 2}:
    276     raise ValueError(
    277         f"IPython won't let you open fd={file} by default "
    278         "as it is likely to crash IPython. If you know what you are doi
ng, "
    279         "you can use builtins' open."
    280     )
--> 282 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'relatorio_jogos.txt'
```

Desafio

No Brasil, existe um órgão responsável por gerar as estatísticas da atividade econômica no país. Para tal tarefa, as atividades são classificadas em grupos; por exemplo, as atividades do grupo 262 referem-se à fabricação de equipamentos de informática e periféricos. "A CNAE, Classificação Nacional de Atividades Econômicas, é a classificação oficialmente adotada pelo Sistema Estatístico Nacional na produção de estatísticas por tipo de atividade econômica, e pela Administração Pública, na identificação da atividade econômica em cadastros e registros de pessoa jurídica." (API CNAE, 2017, [s.p.])

Como desenvolvedor em uma empresa de consultoria de software, você foi alocado em um projeto com base no qual o cliente deseja automatizar a extração dos dados do CNAE e gerar um relatório. Os dados estão disponíveis no endereço https://servicodados.ibge.gov.br/api/v2/cnae/classes. Você deve extraí-los e gerar as seguintes informações:

- Quantas atividades distintas estão registradas?
- Quantos grupos de atividades existem?
- Quantas atividades estão cadastradas em cada grupo?
- Qual grupo ou quais grupos possuem o maior número de atividades vinculadas?

RESOLUÇÃO

Para automatizar o processo de extração dos dados do CNAE e gerar o relatório, vamos ter de usar bibliotecas. Para fazer a extração dos dados do CNAE, podemos usar a biblioteca requests. Para responder às perguntas, vamos precisar manipular listas e dicionários. Então vamos começar pela extração.

In [ ]:
```
import requests

dados = requests.get('https://servicodados.ibge.gov.br/api/v2/cnae/classes').jsc
dados[0]
```

Out[ ]:
```
{'id': '01113',
 'descricao': 'CULTIVO DE CEREAIS',
 'grupo': {'id': '011',
  'descricao': 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
  'divisao': {'id': '01',
   'descricao': 'AGRICULTURA, PECUÁRIA E SERVIÇOS RELACIONADOS',
   'secao': {'id': 'A',
    'descricao': 'AGRICULTURA, PECUÁRIA, PRODUÇÃO FLORESTAL, PESCA E AQÜICULTUR
A'}}},
 'observacoes': ['Esta classe compreende - o cultivo de alpiste, arroz, aveia,
centeio, cevada, milho, milheto, painço, sorgo, trigo, trigo preto, triticale e
outros cereais não especificados anteriormente',
  'Esta classe compreende ainda - o beneficiamento de cereais em estabeleciment
o agrícola, quando atividade complementar ao cultivo\r\n- a produção de semente
s de cereais, quando atividade complementar ao cultivo',
  'Esta classe NÃO compreende - a produção de sementes certificadas dos cereais
desta classe, inclusive modificadas geneticamente (01.41-5)\r\n- os serviços de
preparação de terreno, cultivo e colheita realizados sob contrato (01.61-0)\r\n
- o beneficiamento de cereais em estabelecimento agrícola realizado sob contrat
o (01.63-6)\r\n- o processamento ou beneficiamento de cereais em estabeleciment
o não-agrícola (grupo 10.4) e (grupo 10.6)\r\n- a produção de biocombustível (1
9.32-2)']}
```

In [ ]:
```
#quantidade distintas de atividades, basta saber o tamanho da lista.

qtde_atividades_distintas = len(dados)
print(qtde_atividades_distintas)
```

```
673
```

In [ ]:
```
#criar uma lista dos grupos de atividades, extraindo a descricao

grupos = []
for registro in dados:
    grupos.append(registro['grupo']['descricao'])

grupos[:10] #exibe os 10 primeiros registros
```

Out[ ]:
```
['PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS',
 'HORTICULTURA E FLORICULTURA',
 'HORTICULTURA E FLORICULTURA',
 'EXTRAÇÃO DE MINERAIS METÁLICOS NÃO-FERROSOS']
```

In [ ]:
```
#extraindo a quantidade de grupos de atividades
qtde_grupos_distintos = len(set(grupos)) # set cria uma estrutura de dados remov
print(qtde_grupos_distintos)

grupos_count = [(grupo, grupos.count(grupo)) for grupo in set(grupos)]
grupos_count[:5]
```

```
285
```

Out[ ]:
```
[('ATIVIDADES AUXILIARES DOS SEGUROS, DA PREVIDÊNCIA COMPLEMENTAR E DOS PLANOS
DE SAÚDE',
  3),
 ('FABRICAÇÃO DE BRINQUEDOS E JOGOS RECREATIVOS', 1),
 ('EXTRAÇÃO DE MINÉRIO DE FERRO', 1),
 ('SEDES DE EMPRESAS E UNIDADES ADMINISTRATIVAS LOCAIS', 1),
 ('ARMAZENAMENTO, CARGA E DESCARGA', 2)]
```

In [ ]:
```
#transformando a lista em dicionario
grupos_count = dict(grupos_count)
grupos_count
```

Out[ ]:
{'ATIVIDADES AUXILIARES DOS SEGUROS, DA PREVIDÊNCIA COMPLEMENTAR E DOS PLANOS DE SAÚDE': 3,
 'FABRICAÇÃO DE BRINQUEDOS E JOGOS RECREATIVOS': 1,
 'EXTRAÇÃO DE MINÉRIO DE FERRO': 1,
 'SEDES DE EMPRESAS E UNIDADES ADMINISTRATIVAS LOCAIS': 1,
 'ARMAZENAMENTO, CARGA E DESCARGA': 2,
 'EDUCAÇÃO INFANTIL E ENSINO FUNDAMENTAL': 3,
 'FABRICAÇÃO DE ARTEFATOS DE CONCRETO, CIMENTO, FIBROCIMENTO, GESSO E MATERIAIS SEMELHANTES': 1,
 'FABRICAÇÃO DE MÁQUINAS E EQUIPAMENTOS DE USO INDUSTRIAL ESPECÍFICO': 7,
 'ATIVIDADES IMOBILIÁRIAS POR CONTRATO OU COMISSÃO': 2,
 'TRANSPORTE RODOVIÁRIO DE PASSAGEIROS': 5,
 'COMÉRCIO ATACADISTA DE MÁQUINAS, APARELHOS E EQUIPAMENTOS, EXCETO DE TECNOLOGIAS DE INFORMAÇÃO E COMUNICAÇÃO': 6,
 'ATIVIDADES PROFISSIONAIS, CIENTÍFICAS E TÉCNICAS NÃO ESPECIFICADAS ANTERIORMENTE': 1,
 'FABRICAÇÃO E REFINO DE AÇÚCAR': 2,
 'FABRICAÇÃO DE PRODUTOS FARMACÊUTICOS': 3,
 'OUTROS TRANSPORTES AQUAVIÁRIOS': 2,
 'ATIVIDADES DE ASSOCIAÇÕES DE DEFESA DE DIREITOS SOCIAIS': 1,
 'ATIVIDADES DOS SERVIÇOS DE TECNOLOGIA DA INFORMAÇÃO': 5,
 'COLETA DE RESÍDUOS': 2,
 'FABRICAÇÃO DE CIMENTO': 1,
 'PRODUÇÃO DE TUBOS DE AÇO, EXCETO TUBOS SEM COSTURA': 2,
 'SERVIÇOS DE PRÉ-IMPRESSÃO E ACABAMENTOS GRÁFICOS': 2,
 'EDUCAÇÃO PROFISSIONAL DE NÍVEL TÉCNICO E TECNOLÓGICO': 2,
 'ATIVIDADES DE ATENDIMENTO HOSPITALAR': 1,
 'FABRICAÇÃO DE ARTIGOS PARA VIAGEM E DE ARTEFATOS DIVERSOS DE COURO': 2,
 'FABRICAÇÃO DE MÓVEIS': 4,
 'HOTÉIS E SIMILARES': 1,
 'EDUCAÇÃO SUPERIOR': 3,
 'CAPTAÇÃO, TRATAMENTO E DISTRIBUIÇÃO DE ÁGUA': 1,
 'ABATE E FABRICAÇÃO DE PRODUTOS DE CARNE': 3,
 'TRANSPORTE AÉREO DE PASSAGEIROS': 2,
 'SEGURIDADE SOCIAL OBRIGATÓRIA': 1,
 'FABRICAÇÃO DE MÁQUINAS-FERRAMENTA': 1,
 'FABRICAÇÃO DE EQUIPAMENTOS PARA DISTRIBUIÇÃO E CONTROLE DE ENERGIA ELÉTRICA': 3,
 'FABRICAÇÃO DE ELETRODOMÉSTICOS': 2,
 'DEMOLIÇÃO E PREPARAÇÃO DO TERRENO': 4,
 'SERVIÇOS DE CATERING, BUFÊ E OUTROS SERVIÇOS DE COMIDA PREPARADA': 1,
 'ACABAMENTOS EM FIOS, TECIDOS E ARTEFATOS TÊXTEIS': 1,
 'ATIVIDADES ARTÍSTICAS, CRIATIVAS E DE ESPETÁCULOS': 3,
 'EXTRAÇÃO DE OUTROS MINERAIS NÃO-METÁLICOS': 4,
 'FABRICAÇÃO DE OUTROS PRODUTOS ALIMENTÍCIOS': 7,
 'FABRICAÇÃO DE EQUIPAMENTOS DE INFORMÁTICA E PERIFÉRICOS': 2,
 'REPRESENTANTES COMERCIAIS E AGENTES DO COMÉRCIO, EXCETO DE VEÍCULOS AUTOMOTORES E MOTOCICLETAS': 9,
 'COMÉRCIO DE PEÇAS E ACESSÓRIOS PARA VEÍCULOS AUTOMOTORES': 1,
 'FABRICAÇÃO DE PRODUTOS DE MADEIRA, CORTIÇA E MATERIAL TRANÇADO, EXCETO MÓVEIS': 4,
 'OUTROS TIPOS DE ALOJAMENTO NÃO ESPECIFICADOS ANTERIORMENTE': 1,
 'OBRAS DE ACABAMENTO': 1,
 'PREVIDÊNCIA COMPLEMENTAR': 2,
 'GERAÇÃO, TRANSMISSÃO E DISTRIBUIÇÃO DE ENERGIA ELÉTRICA': 4,
 'TRATAMENTO E DISPOSIÇÃO DE RESÍDUOS': 2,
 'RECONDICIONAMENTO E RECUPERAÇÃO DE MOTORES PARA VEÍCULOS AUTOMOTORES': 1,
 'INCORPORAÇÃO DE EMPREENDIMENTOS IMOBILIÁRIOS': 1,
 'OPERADORAS DE TELEVISÃO POR ASSINATURA': 3,
 'FABRICAÇÃO DE EQUIPAMENTOS DE TRANSPORTE NÃO ESPECIFICADOS ANTERIORMENTE': 3,

```
 'CAÇA E SERVIÇOS RELACIONADOS': 1,
 'FABRICAÇÃO DE ÓLEOS E GORDURAS VEGETAIS E ANIMAIS': 3,
 'FABRICAÇÃO DE CALÇADOS': 4,
 'COMÉRCIO VAREJISTA DE PRODUTOS FARMACÊUTICOS, PERFUMARIA E COSMÉTICOS E ARTIG
OS MÉDICOS, ÓPTICOS E ORTOPÉDICOS': 4,
 'TELECOMUNICAÇÕES SEM FIO': 1,
 'OBRAS DE INFRA-ESTRUTURA PARA ENERGIA ELÉTRICA, TELECOMUNICAÇÕES, ÁGUA, ESGOT
O E TRANSPORTE POR DUTOS': 3,
 'ATIVIDADES DE SOCIEDADES DE PARTICIPAÇÃO': 3,
 'ATIVIDADES DE ASSISTÊNCIA SOCIAL PRESTADAS EM RESIDÊNCIAS COLETIVAS E PARTICU
LARES': 1,
 'LOCAÇÃO DE MÃO-DE-OBRA TEMPORÁRIA': 1,
 'FABRICAÇÃO DE PRODUTOS DE MATERIAL PLÁSTICO': 4,
 'HORTICULTURA E FLORICULTURA': 2,
 'OUTROS SERVIÇOS ESPECIALIZADOS PARA CONSTRUÇÃO': 2,
 'SEGUROS-SAÚDE': 1,
 'ATIVIDADES LIGADAS AO PATRIMÔNIO CULTURAL E AMBIENTAL': 3,
 'COMÉRCIO VAREJISTA DE ARTIGOS CULTURAIS, RECREATIVOS E ESPORTIVOS': 3,
 'ATIVIDADES VETERINÁRIAS': 1,
 'EDIÇÃO INTEGRADA À IMPRESSÃO DE LIVROS, JORNAIS, REVISTAS E OUTRAS PUBLICAÇÕE
S': 4,
 'CONSTRUÇÃO DE EDIFÍCIOS': 1,
 'FABRICAÇÃO DE PRODUTOS CERÂMICOS': 3,
 'PRODUÇÃO DE LAVOURAS TEMPORÁRIAS': 7,
 'FABRICAÇÃO DE EQUIPAMENTO BÉLICO PESADO, ARMAS E MUNIÇÕES': 1,
 'FABRICAÇÃO DE ARTEFATOS TÊXTEIS, EXCETO VESTUÁRIO': 5,
 'FABRICAÇÃO DE PAPEL, CARTOLINA E PAPEL-CARTÃO': 2,
 'FABRICAÇÃO DE MOTORES, BOMBAS, COMPRESSORES E EQUIPAMENTOS DE TRANSMISSÃO':
5,
 'COMÉRCIO VAREJISTA NÃO-ESPECIALIZADO': 3,
 'OUTRAS ATIVIDADES DE SERVIÇOS PRESTADOS PRINCIPALMENTE ÀS EMPRESAS': 3,
 'SIDERURGIA': 4,
 'TRANSPORTE MARÍTIMO DE CABOTAGEM E LONGO CURSO': 2,
 'AQÜICULTURA': 2,
 'FABRICAÇÃO DE PRODUTOS DERIVADOS DO PETRÓLEO': 2,
 'METALURGIA DOS METAIS NÃO-FERROSOS': 4,
 'SERVIÇOS DE RESERVAS E OUTROS SERVIÇOS DE TURISMO NÃO ESPECIFICADOS ANTERIORM
ENTE': 1,
 'ATIVIDADES AUXILIARES DOS TRANSPORTES AQUAVIÁRIOS': 3,
 'FABRICAÇÃO DE RESINAS E ELASTÔMEROS': 3,
 'INTERMEDIAÇÃO MONETÁRIA - DEPÓSITOS À VISTA': 4,
 'PRODUÇÃO E DISTRIBUIÇÃO DE VAPOR, ÁGUA QUENTE E AR CONDICIONADO': 1,
 'ATIVIDADES DE ADMINISTRAÇÃO DE FUNDOS POR CONTRATO OU COMISSÃO': 1,
 'ALUGUEL DE OBJETOS PESSOAIS E DOMÉSTICOS': 4,
 'ATIVIDADES DE SERVIÇOS FINANCEIROS NÃO ESPECIFICADAS ANTERIORMENTE': 4,
 'AGÊNCIAS DE VIAGENS E OPERADORES TURÍSTICOS': 2,
 'COQUERIAS': 1,
 'FABRICAÇÃO DE TINTAS, VERNIZES, ESMALTES, LACAS E PRODUTOS AFINS': 3,
 'FABRICAÇÃO DE APARELHOS DE RECEPÇÃO, REPRODUÇÃO, GRAVAÇÃO E AMPLIFICAÇÃO DE Á
UDIO E VÍDEO': 1,
 'FABRICAÇÃO DE EQUIPAMENTOS E APARELHOS ELÉTRICOS NÃO ESPECIFICADOS ANTERIORME
NTE': 1,
 'TRANSPORTE DUTOVIÁRIO': 1,
 'FABRICAÇÃO DE INSTRUMENTOS MUSICAIS': 1,
 'SOCIEDADES DE CAPITALIZAÇÃO': 1,
 'FABRICAÇÃO DE MÁQUINAS E EQUIPAMENTOS DE USO NA EXTRAÇÃO MINERAL E NA CONSTRU
ÇÃO': 4,
 'DESCONTAMINAÇÃO E OUTROS SERVIÇOS DE GESTÃO DE RESÍDUOS': 1,
 'FABRICAÇÃO DE PRODUTOS DE BORRACHA': 3,
 'PESQUISAS DE MERCADO E DE OPINIÃO PÚBLICA': 1,
```

```
 'ATIVIDADES DE APOIO À EXTRAÇÃO DE MINERAIS, EXCETO PETRÓLEO E GÁS NATURAL':
1,
 'PROCESSAMENTO INDUSTRIAL DO FUMO': 1,
 'ATIVIDADES DE VIGILÂNCIA, SEGURANÇA PRIVADA E TRANSPORTE DE VALORES': 2,
 'ADMINISTRAÇÃO DO ESTADO E DA POLÍTICA ECONÔMICA E SOCIAL': 3,
 'ATIVIDADES DE SERVIÇOS DE COMPLEMENTAÇÃO DIAGNÓSTICA E TERAPÊUTICA': 1,
 'TESTES E ANÁLISES TÉCNICAS': 1,
 'PESCA': 2,
 'ATIVIDADES DE CONTABILIDADE, CONSULTORIA E AUDITORIA CONTÁBIL E TRIBUTÁRIA':
1,
 'GESTÃO DE ATIVOS INTANGÍVEIS NÃO-FINANCEIROS': 1,
 'FABRICAÇÃO DE EMBALAGENS DE PAPEL, CARTOLINA, PAPEL-CARTÃO E PAPELÃO ONDULAD
O': 3,
 'PREPARAÇÃO E FIAÇÃO DE FIBRAS TÊXTEIS': 4,
 'ATIVIDADES PAISAGÍSTICAS': 1,
 'FABRICAÇÃO DE VEÍCULOS MILITARES DE COMBATE': 1,
 'TRANSPORTE AÉREO DE CARGA': 1,
 'FABRICAÇÃO DE CAMINHÕES E ÔNIBUS': 1,
 'COMÉRCIO VAREJISTA DE MATERIAL DE CONSTRUÇÃO': 4,
 'ATIVIDADES DE PROFISSIONAIS DA ÁREA DE SAÚDE, EXCETO MÉDICOS E ODONTÓLOGOS':
1,
 'RECUPERAÇÃO DE MATERIAIS': 3,
 'ATIVIDADES DE ORGANIZAÇÕES ASSOCIATIVAS PATRONAIS, EMPRESARIAIS E PROFISSIONA
IS': 2,
 'TRENS TURÍSTICOS, TELEFÉRICOS E SIMILARES': 1,
 'FABRICAÇÃO DE INSTRUMENTOS E MATERIAIS PARA USO MÉDICO E ODONTOLÓGICO E DE AR
TIGOS ÓPTICOS': 1,
 'FABRICAÇÃO DE PRODUTOS FARMOQUÍMICOS': 1,
 'ATIVIDADES DE CORREIO': 1,
 'COMÉRCIO ATACADISTA DE MADEIRA, FERRAGENS, FERRAMENTAS, MATERIAL ELÉTRICO E M
ATERIAL DE CONSTRUÇÃO': 5,
 'FABRICAÇÃO DE TECIDOS DE MALHA': 1,
 'CONSTRUÇÃO DE RODOVIAS, FERROVIAS, OBRAS URBANAS E OBRAS-DE-ARTE ESPECIAIS':
3,
 'ATIVIDADES DE MONITORAMENTO DE SISTEMAS DE SEGURANÇA': 1,
 'ATIVIDADES CINEMATOGRÁFICAS, PRODUÇÃO DE VÍDEOS E DE PROGRAMAS DE TELEVISÃO':
4,
 'EXTRAÇÃO DE PEDRA, AREIA E ARGILA': 1,
 'CURTIMENTO E OUTRAS PREPARAÇÕES DE COURO': 1,
 'DESDOBRAMENTO DE MADEIRA': 1,
 'ATIVIDADES DE MALOTE E DE ENTREGA': 1,
 'ATIVIDADES ESPORTIVAS': 4,
 'FABRICAÇÃO DE ARTIGOS DE JOALHERIA, BIJUTERIA E SEMELHANTES': 2,
 'FABRICAÇÃO DE AERONAVES': 2,
 'EXTRAÇÃO DE CARVÃO MINERAL': 1,
 'FABRICAÇÃO DE APARELHOS ELETROMÉDICOS E ELETROTERAPÊUTICOS E EQUIPAMENTOS DE
IRRADIAÇÃO': 1,
 'FABRICAÇÃO DE PEÇAS E ACESSÓRIOS PARA VEÍCULOS AUTOMOTORES': 6,
 'EDIÇÃO DE LIVROS, JORNAIS, REVISTAS E OUTRAS ATIVIDADES DE EDIÇÃO': 4,
 'EXTRAÇÃO DE MINERAIS METÁLICOS NÃO-FERROSOS': 6,
 'OUTRAS ATIVIDADES DE ENSINO': 4,
 'COMÉRCIO DE VEÍCULOS AUTOMOTORES': 2,
 'ATIVIDADES DE ASSISTÊNCIA A IDOSOS, DEFICIENTES FÍSICOS, IMUNODEPRIMIDOS E CO
NVALESCENTES, E DE INFRA-ESTRUTURA E APOIO A PACIENTES PRESTADAS EM RESIDÊNCIAS
COLETIVAS E PARTICULARES': 2,
 'FABRICAÇÃO DE VEÍCULOS FERROVIÁRIOS': 2,
 'TELECOMUNICAÇÕES POR FIO': 1,
 'ATIVIDADES JURÍDICAS': 2,
 'FABRICAÇÃO DE PILHAS, BATERIAS E ACUMULADORES ELÉTRICOS': 2,
 'SERVIÇOS DE ESCRITÓRIO E APOIO ADMINISTRATIVO': 2,
```

```
 'FABRICAÇÃO DE CONSERVAS DE FRUTAS, LEGUMES E OUTROS VEGETAIS': 3,
 'FABRICAÇÃO DE TRATORES E DE MÁQUINAS E EQUIPAMENTOS PARA A AGRICULTURA E PECU
ÁRIA': 3,
 'ATIVIDADES AUXILIARES DOS TRANSPORTES AÉREOS': 1,
 'SERVIÇOS COMBINADOS PARA APOIO A EDIFÍCIOS': 2,
 'ATIVIDADES DE ATENÇÃO AMBULATORIAL EXECUTADAS POR MÉDICOS E ODONTÓLOGOS': 1,
 'RESSEGUROS': 1,
 'FABRICAÇÃO DE AUTOMÓVEIS, CAMIONETAS E UTILITÁRIOS': 1,
 'ATIVIDADES DE CONSULTORIA EM GESTÃO EMPRESARIAL': 1,
 'CONSTRUÇÃO DE OUTRAS OBRAS DE INFRA-ESTRUTURA': 3,
 'FABRICAÇÃO DE PRODUTOS DIVERSOS DE PAPEL, CARTOLINA, PAPEL-CARTÃO E PAPELÃO O
NDULADO': 3,
 'ALUGUEL DE MÁQUINAS E EQUIPAMENTOS SEM OPERADOR': 4,
 'ATIVIDADES DE ORGANIZAÇÕES ASSOCIATIVAS NÃO ESPECIFICADAS ANTERIORMENTE': 4,
 'ATIVIDADES DE GRAVAÇÃO DE SOM E DE EDIÇÃO DE MÚSICA': 1,
 'TELECOMUNICAÇÕES POR SATÉLITE': 1,
 'OUTRAS ATIVIDADES DE SERVIÇOS PESSOAIS': 4,
 'MOAGEM, FABRICAÇÃO DE PRODUTOS AMILÁCEOS E DE ALIMENTOS PARA ANIMAIS': 7,
 'FABRICAÇÃO DE TANQUES, RESERVATÓRIOS METÁLICOS E CALDEIRAS': 2,
 'COMÉRCIO ATACADISTA ESPECIALIZADO EM PRODUTOS ALIMENTÍCIOS, BEBIDAS E FUMO':
8,
 'ATIVIDADES DE APOIO À PRODUÇÃO FLORESTAL': 1,
 'FABRICAÇÃO DE EQUIPAMENTOS DE COMUNICAÇÃO': 2,
 'FABRICAÇÃO DE BEBIDAS ALCOÓLICAS': 3,
 'ATIVIDADES DE APOIO À GESTÃO DE SAÚDE': 1,
 'ATIVIDADES FOTOGRÁFICAS E SIMILARES': 1,
 'COMÉRCIO ATACADISTA DE MATÉRIAS-PRIMAS AGRÍCOLAS E ANIMAIS VIVOS': 3,
 'ATIVIDADES AUXILIARES DOS TRANSPORTES TERRESTRES': 4,
 'SEGUROS DE VIDA E NÃO-VIDA': 2,
 'FABRICAÇÃO DE VIDRO E DE PRODUTOS DO VIDRO': 3,
 'BANCO CENTRAL': 1,
 'FABRICAÇÃO DE BEBIDAS NÃO-ALCOÓLICAS': 2,
 'ATIVIDADE DE IMPRESSÃO': 3,
 'MANUTENÇÃO E REPARAÇÃO DE VEÍCULOS AUTOMOTORES': 1,
 'FORJARIA, ESTAMPARIA, METALURGIA DO PÓ E SERVIÇOS DE TRATAMENTO DE METAIS':
3,
 'FABRICAÇÃO DE ARTIGOS DE CUTELARIA, DE SERRALHERIA E FERRAMENTAS': 3,
 'COMÉRCIO VAREJISTA DE PRODUTOS NOVOS NÃO ESPECIFICADOS ANTERIORMENTE E DE PRO
DUTOS USADOS': 6,
 'PRODUÇÃO DE SEMENTES E MUDAS CERTIFICADAS': 2,
 'PRODUÇÃO DE FERRO-GUSA E DE FERROLIGAS': 2,
 'RESTAURANTES E OUTROS SERVIÇOS DE ALIMENTAÇÃO E BEBIDAS': 2,
 'ATIVIDADES DE APOIO À AGRICULTURA E À PECUÁRIA; ATIVIDADES DE PÓS-COLHEITA':
3,
 'TRANSPORTE ESPACIAL': 1,
 'FABRICAÇÃO DE DEFENSIVOS AGRÍCOLAS E DESINFESTANTES DOMISSANITÁRIOS': 2,
 'FABRICAÇÃO DE BIOCOMBUSTÍVEIS': 2,
 'ATIVIDADES DE RÁDIO': 1,
 'OUTRAS ATIVIDADES DE PRESTAÇÃO DE SERVIÇOS DE INFORMAÇÃO': 2,
 'FABRICAÇÃO DE PRODUTOS QUÍMICOS ORGÂNICOS': 3,
 'SERVIÇOS DE ARQUITETURA E ENGENHARIA E ATIVIDADES TÉCNICAS RELACIONADAS': 3,
 'SERVIÇOS DE ASSISTÊNCIA SOCIAL SEM ALOJAMENTO': 1,
 'EXTRAÇÃO DE PETRÓLEO E GÁS NATURAL': 1,
 'FABRICAÇÃO DE CELULOSE E OUTRAS PASTAS PARA A FABRICAÇÃO DE PAPEL': 1,
 'FABRICAÇÃO DE ARTIGOS DE MALHARIA E TRICOTAGEM': 2,
 'FABRICAÇÃO DE PARTES PARA CALÇADOS, DE QUALQUER MATERIAL': 1,
 'ORGANISMOS INTERNACIONAIS E OUTRAS INSTITUIÇÕES EXTRATERRITORIAIS': 1,
 'ATIVIDADES DE RECREAÇÃO E LAZER': 2,
 'LATICÍNIOS': 3,
 'COMÉRCIO ATACADISTA ESPECIALIZADO EM OUTROS PRODUTOS': 8,
```

```
 'FABRICAÇÃO DE EQUIPAMENTOS E INSTRUMENTOS ÓPTICOS, FOTOGRÁFICOS E CINEMATOGRÁ
FICOS': 1,
 'PUBLICIDADE': 3,
 'ARRENDAMENTO MERCANTIL': 1,
 'REPRODUÇÃO DE MATERIAIS GRAVADOS EM QUALQUER SUPORTE': 1,
 'CONSTRUÇÃO DE EMBARCAÇÕES': 2,
 'ATIVIDADES RELACIONADAS À ORGANIZAÇÃO DO TRANSPORTE DE CARGA': 1,
 'FUNDIÇÃO': 2,
 'FABRICAÇÃO DE COMPONENTES ELETRÔNICOS': 1,
 'PRESERVAÇÃO DO PESCADO E FABRICAÇÃO DE PRODUTOS DO PESCADO': 1,
 'PLANOS DE SAÚDE': 1,
 'SELEÇÃO E AGENCIAMENTO DE MÃO-DE-OBRA': 1,
 'FORNECIMENTO E GESTÃO DE RECURSOS HUMANOS PARA TERCEIROS': 1,
 'ESGOTO E ATIVIDADES RELACIONADAS': 2,
 'REPARAÇÃO E MANUTENÇÃO DE EQUIPAMENTOS DE INFORMÁTICA E COMUNICAÇÃO': 2,
 'FABRICAÇÃO DE APARELHOS E INSTRUMENTOS DE MEDIDA, TESTE E CONTROLE; CRONÔMETR
OS E RELÓGIOS': 2,
 'FABRICAÇÃO DE LÂMPADAS E OUTROS EQUIPAMENTOS DE ILUMINAÇÃO': 1,
 'ATIVIDADES DE TELEVISÃO': 2,
 'FABRICAÇÃO DE MÍDIAS VIRGENS, MAGNÉTICAS E ÓPTICAS': 1,
 'OUTRAS ATIVIDADES DE TELECOMUNICAÇÕES': 1,
 'ATIVIDADES DE ORGANIZAÇÃO DE EVENTOS, EXCETO CULTURAIS E ESPORTIVOS': 1,
 'FABRICAÇÃO DE CABINES, CARROCERIAS E REBOQUES PARA VEÍCULOS AUTOMOTORES': 1,
 'TRANSPORTE RODOVIÁRIO DE CARGA': 1,
 'REPARAÇÃO E MANUTENÇÃO DE OBJETOS E EQUIPAMENTOS PESSOAIS E DOMÉSTICOS': 2,
 'ATIVIDADES DE ORGANIZAÇÕES SINDICAIS': 1,
 'TRANSPORTE POR NAVEGAÇÃO INTERIOR': 2,
 'COMÉRCIO ATACADISTA DE PRODUTOS DE CONSUMO NÃO-ALIMENTAR': 8,
 'SERVIÇOS COLETIVOS PRESTADOS PELA ADMINISTRAÇÃO PÚBLICA': 5,
 'SERVIÇOS DOMÉSTICOS': 1,
 'FABRICAÇÃO DE GERADORES, TRANSFORMADORES E MOTORES ELÉTRICOS': 1,
 'FABRICAÇÃO DE PRODUTOS DO FUMO': 1,
 'ATIVIDADES DE ATENÇÃO À SAÚDE HUMANA NÃO ESPECIFICADAS ANTERIORMENTE': 1,
 'NAVEGAÇÃO DE APOIO': 1,
 'ATIVIDADES DE LIMPEZA': 3,
 'COMÉRCIO VAREJISTA DE PRODUTOS ALIMENTÍCIOS, BEBIDAS E FUMO': 5,
 'COMÉRCIO, MANUTENÇÃO E REPARAÇÃO DE MOTOCICLETAS, PEÇAS E ACESSÓRIOS': 3,
 'FABRICAÇÃO DE PRODUTOS DIVERSOS': 3,
 'ATIVIDADES DE APOIO À EDUCAÇÃO': 1,
 'PRODUÇÃO DE LAVOURAS PERMANENTES': 6,
 'LOCAÇÃO DE MEIOS DE TRANSPORTE SEM CONDUTOR': 2,
 'FABRICAÇÃO DE ESTRUTURAS METÁLICAS E OBRAS DE CALDEIRARIA PESADA': 3,
 'ATIVIDADES IMOBILIÁRIAS DE IMÓVEIS PRÓPRIOS': 1,
 'ATIVIDADES DE INVESTIGAÇÃO PARTICULAR': 1,
 'PRODUÇÃO FLORESTAL - FLORESTAS PLANTADAS': 1,
 'TRANSPORTE FERROVIÁRIO E METROFERROVIÁRIO': 2,
 'PECUÁRIA': 6,
 'COMÉRCIO VAREJISTA DE EQUIPAMENTOS DE INFORMÁTICA E COMUNICAÇÃO; EQUIPAMENTOS
E ARTIGOS DE USO DOMÉSTICO': 8,
 'PESQUISA E DESENVOLVIMENTO EXPERIMENTAL EM CIÊNCIAS SOCIAIS E HUMANAS': 1,
 'FABRICAÇÃO DE PRODUTOS E PREPARADOS QUÍMICOS DIVERSOS': 5,
 'PRODUÇÃO E DISTRIBUIÇÃO DE COMBUSTÍVEIS GASOSOS POR REDES URBANAS': 1,
 'COMÉRCIO AMBULANTE E OUTROS TIPOS DE COMÉRCIO VAREJISTA': 1,
 'ENSINO MÉDIO': 1,
 'TORREFAÇÃO E MOAGEM DE CAFÉ': 2,
 'CONFECÇÃO DE ARTIGOS DO VESTUÁRIO E ACESSÓRIOS': 4,
 'FABRICAÇÃO DE PRODUTOS DE METAL NÃO ESPECIFICADOS ANTERIORMENTE': 4,
 'INSTALAÇÕES ELÉTRICAS, HIDRÁULICAS E OUTRAS INSTALAÇÕES EM CONSTRUÇÕES': 3,
 'FABRICAÇÃO DE ARTEFATOS PARA PESCA E ESPORTE': 1,
 'TRATAMENTO DE DADOS, HOSPEDAGEM NA INTERNET E OUTRAS ATIVIDADES RELACIONADA
```

```
S': 2,
 'FABRICAÇÃO DE SABÕES, DETERGENTES, PRODUTOS DE LIMPEZA, COSMÉTICOS, PRODUTOS
DE PERFUMARIA E DE HIGIENE PESSOAL': 3,
 'COMÉRCIO ATACADISTA NÃO-ESPECIALIZADO': 3,
 'SERVIÇOS MÓVEIS DE ATENDIMENTO A URGÊNCIAS E DE REMOÇÃO DE PACIENTES': 2,
 'ATIVIDADES AUXILIARES DOS SERVIÇOS FINANCEIROS': 4,
 'ATIVIDADES DE TELEATENDIMENTO': 1,
 'TECELAGEM, EXCETO MALHA': 3,
 'FABRICAÇÃO DE FIBRAS ARTIFICIAIS E SINTÉTICAS': 1,
 'PRODUÇÃO FLORESTAL - FLORESTAS NATIVAS': 1,
 'FABRICAÇÃO DE PRODUTOS QUÍMICOS INORGÂNICOS': 5,
 'INSTALAÇÃO DE MÁQUINAS E EQUIPAMENTOS': 2,
 'INTERMEDIAÇÃO NÃO-MONETÁRIA - OUTROS INSTRUMENTOS DE CAPTAÇÃO': 8,
 'ATIVIDADES DE APOIO À EXTRAÇÃO DE PETRÓLEO E GÁS NATURAL': 1,
 'APARELHAMENTO DE PEDRAS E FABRICAÇÃO DE OUTROS PRODUTOS DE MINERAIS NÃO-METÁL
ICOS': 3,
 'ATIVIDADES DE ASSISTÊNCIA PSICOSSOCIAL E À SAÚDE A PORTADORES DE DISTÚRBIOS P
SÍQUICOS, DEFICIÊNCIA MENTAL E DEPENDÊNCIA QUÍMICA': 1,
 'COMÉRCIO VAREJISTA DE COMBUSTÍVEIS PARA VEÍCULOS AUTOMOTORES': 2,
 'COMÉRCIO ATACADISTA DE EQUIPAMENTOS E PRODUTOS DE TECNOLOGIAS DE INFORMAÇÃO E
COMUNICAÇÃO': 2,
 'ATIVIDADES DE EXPLORAÇÃO DE JOGOS DE AZAR E APOSTAS': 1,
 'DESIGN E DECORAÇÃO DE INTERIORES': 1,
 'PESQUISA E DESENVOLVIMENTO EXPERIMENTAL EM CIÊNCIAS FÍSICAS E NATURAIS': 1,
 'MANUTENÇÃO E REPARAÇÃO DE MÁQUINAS E EQUIPAMENTOS': 8,
 'FABRICAÇÃO DE MÁQUINAS E EQUIPAMENTOS DE USO GERAL': 6,
 'FUNDOS DE INVESTIMENTO': 1}
```

In [ ]:
```
# descobrindo quais grupos possuem mais atividades
valor_maximo = max(grupos_count.values())
grupos_mais_atividades = [chave for (chave, valor) in grupos_count.items() if va
print(len(grupos_mais_atividades))
grupos_mais_atividades
```

```
1
```

Out[ ]:
```
['REPRESENTANTES COMERCIAIS E AGENTES DO COMÉRCIO, EXCETO DE VEÍCULOS AUTOMOTOR
ES E MOTOCICLETAS']
```

In [ ]:
```
import requests

from datetime import datetime


class ETL:
    def __init__(self):
        self.url = None

    def extract_cnae_data(self, url):
        self.url = url
        data_extracao = datetime.today().strftime("%Y/%m/%d - %H:%M:%S")
        # Faz extração
        dados = requests.get(self.url).json()

        # Extrai os grupos dos registros
        grupos = []
        for registro in dados:
            grupos.append(registro['grupo']['descricao'])

        # Cria uma lista de tuplas (grupo, quantidade_atividades)
        grupos_count = [(grupo, grupos.count(grupo)) for grupo in set(grupos)]
```

```
        grupos_count = dict(grupos_count)  # transforma a lista em dicionário

        valor_maximo = max(grupos_count.values())  # Captura o valor máximo de a
        # Gera uma lista com os grupos que possuem a quantidade máxima de ativia
        grupos_mais_atividades = [chave for (chave, valor) in grupos_count.items

        # informações
        qtde_atividades_distintas = len(dados)
        qtde_grupos_distintos = len(set(grupos))

        print(f"Dados extraídos em: {data_extracao}")
        print(f"Quantidade de atividades distintas = {qtde_atividades_distintas}
        print(f"Quantidade de grupos distintos = {qtde_grupos_distintos}")
        print(f"Grupos com o maior número de atividades = {grupos_mais_atividade

        return None
```

In [ ]:
```
# Usando a classe ETL

ETL().extract_cnae_data('https://servicodados.ibge.gov.br/api/v2/cnae/classes')
```

```
Dados extraídos em: 2023/03/14 - 20:15:26
Quantidade de atividades distintas = 673
Quantidade de grupos distintos = 285
Grupos com o maior número de atividades = ['REPRESENTANTES COMERCIAIS E AGENTES
DO COMÉRCIO, EXCETO DE VEÍCULOS AUTOMOTORES E MOTOCICLETAS'], atividades = 9
```

Aplicação de banco de dados com python

LINGUAGEM DE CONSULTA ESTRUTURADA - SQL

NTRODUÇÃO A BANCO DE DADOS

Grande parte dos softwares que são desenvolvidos (se não todos) acessa algum tipo de mecanismo para armazenar dados. Podem ser dados da aplicação, como cadastro de clientes, ou então dados sobre a execução da solução, os famosos logs. Esses dados podem ser armazenados em arquivos, cenário no qual se destacam os arquivos delimitados, com extensão CSV (comma separated values), os arquivos JSON (JavaScript Object Notation) e os arquivos XML (Extensible Markup Language). Outra opção para persistir os dados é utilizar um sistema de banco de dados.

CONEXÃO COM BANCO DE DADOS RELACIONAL

Ao criar uma aplicação em uma linguagem de programação que precisa acessar um sistema gerenciador de banco de dados relacional (RDBMS), uma vez que são processos distintos, é preciso criar uma conexão entre eles. Após estabelecida a conexão, é possível (de alguma forma) enviar comandos SQL para efetuar as ações no banco (RAMAKRISHNAN; GEHRKE, 2003). Para fazer a conexão e permitir que uma linguagem de programação se comunique com um banco de dados com a utilização da linguagem SQL, podemos usar as tecnologias ODBC (Open Database Connectivity) e JDBC (Java Database Connectivity).

CONEXÃO DE BANCO DADOS SQL EM PYTHON

Agora que já sabemos que para acessar esse tipo de tecnologia precisamos de um mecanismo de conexão (ODBC ou JDBC) e uma linguagem para nos comunicarmos com ele (SQL), vamos ver como atuar em Python.

Para se comunicar com um RDBMS em Python, podemos utilizar bibliotecas já disponíveis, com uso das quais, por meio do driver de um determinado fornecedor, será possível fazer a conexão e a execução de comandos SQL no banco. Por exemplo, para se conectar com um banco de dados Oracle, podemos usar a biblioteca cx-Oracle, ou, para se conectar a um PostgreSQL, temos como opção o psycopg2. Visando à padronização entre todos os módulos de conexão com um RDBMS e o envio de comandos, o PEP 249 (Python Database API Specification v2.0) elenca um conjunto de regras que os fornecedores devem seguir na construção de módulos para acesso a banco de dados. Por exemplo, a documentação diz que todos módulos devem implementar o método connect(parameters...) para se conectar a um banco.

BANCO DE DADOS SQLITE

Essa tecnologia pode ser embutida em telefones celulares e computadores e vem incluída em inúmeros outros aplicativos que as pessoas usam todos os dias. Ao passo que a maioria dos bancos de dados SQL usa um servidor para rodar e gerenciar, o SQLite não possui um processo de servidor separado. O SQLite lê e grava diretamente em arquivos de disco, ou seja, um banco de dados SQL completo com várias tabelas, índices, triggers e visualizações está contido em um único arquivo de disco.

O interpretador Python possui o módulo built-in sqlite3, que permite utilizar o mecanismo de banco de dados SQLite.

CRIANDO UM BANCO DE DADOS

O primeiro passo é importar o módulo sqlite3. Como o módulo está baseado na especificação DB-API 2.0 descrita pelo PEP 249, ele utiliza o método connnect() para se conectar a um banco de dados. Em razão da natureza do SQLite (ser um arquivo no disco rígido), ao nos conectarmos a um banco, o arquivo é imediatamente criado na pasta do projeto (se estiver usando o projeto Anaconda, o arquivo é criado na mesma pasta em que está o Jupyter Notebook). Se desejar criar o arquivo em outra pasta, basta especificar o caminho juntamente com o nome, por exemplo: C:/Users/Documents/meu_projeto/meus_bancos/bancoDB.db. Observe do código a seguir.

In [ ]:
```
import sqlite3

conn = sqlite3.connect('aulaDB.db')
print(type(conn))
```

```
<class 'sqlite3.Connection'>
```

In [ ]:
```
ddl_create = """
  CREATE TABLE fornecedor (
    id_fornecedor INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    nome_fornecedor TEXT NOT NULL,
```

```
    cnpj VARCHAR(18) NOT NULL,
    cidade TEXT,
    estado VARCHAR(2) NOT NULL,
    cep VARCHAR(9) NOT NULL,
    data_cadastro DATE NOT NULL
);
"""

cursor = conn.cursor()
cursor.execute(ddl_create)
print(type(cursor))

print("Tabela criada!")
print("Descrição do cursor: ", cursor.description)
print("Linhas afetadas: ", cursor.rowcount)
cursor.close()
conn.close()
```

```
<class 'sqlite3.Cursor'>
Tabela criada!
Descrição do cursor:  None
Linhas afetadas:  -1
```

CRUD - CREATE, READ, UPDATE, DELETE

CRUD é um acrônimo para as quatro operações de DML que podemos fazer em uma tabela no banco de dados. Podemos inserir informações (create), ler (read), atualizar (update) e apagar (delete). Os passos necessários para efetuar uma das operações do CRUD são sempre os mesmos: (i) estabelecer a conexão com um banco; (ii) criar um cursor e executar o comando; (iii) gravar a operação; (iv) fechar o cursor e a conexão.

In [ ]:
```
# Só é preciso importar a biblioteca uma vez. Importamos novamente para manter t
import sqlite3

conn = sqlite3.connect('aulaDB.db')
cursor = conn.cursor()

cursor.execute("""
INSERT INTO fornecedor (nome_fornecedor, cnpj, cidade, estado, cep, data_cadastr
VALUES ('Empresa A', '11.111.111/1111-11', 'São Paulo', 'SP', '11111-111', '2020
""")

cursor.execute("""
INSERT INTO fornecedor (nome_fornecedor, cnpj, cidade, estado, cep, data_cadastr
VALUES ('Empresa B', '22.222.222/2222-22', 'Rio de Janeiro', 'RJ', '22222-222',
""")

cursor.execute("""
INSERT INTO fornecedor (nome_fornecedor, cnpj, cidade, estado, cep, data_cadastr
VALUES ('Empresa C', '33.333.333/3333-33', 'Curitiba', 'PR', '33333-333', '2020-
""")

conn.commit()

print("Dados inseridos!")
print("Descrição do cursor: ", cursor.description)
print("Linhas afetadas: ", cursor.rowcount)
```

```
cursor.close()
conn.close()
```

```
Dados inseridos!
Descrição do cursor:  None
Linhas afetadas:  1
```

READ

Agora que temos dados na tabela fornecedor, podemos avançar para a segunda operação, que é recuperar os dados. Também precisamos estabelecer uma conexão e criar um objeto cursor para executar a instrução de seleção. Ao executar a seleção, podemos usar o método fetchall() para capturar todas as linhas, através de uma lista de tuplas. Observe o código a seguir.

In [ ]:
```
import sqlite3

conn = sqlite3.connect('aulaDB.db')
cursor = conn.cursor()

cursor.execute("SELECT * FROM fornecedor")
resultado = cursor.fetchall()

print("Descrição do cursor: ", cursor.description)
print("Linhas afetadas: ", cursor.rowcount)

resultado[:2]
```

```
Descrição do cursor:  (('id_fornecedor', None, None, None, None, None, None),
('nome_fornecedor', None, None, None, None, None, None), ('cnpj', None, None, N
one, None, None, None), ('cidade', None, None, None, None, None, None), ('estad
o', None, None, None, None, None, None), ('cep', None, None, None, None, None,
None), ('data_cadastro', None, None, None, None, None, None))
Linhas afetadas:  -1
```

Out[ ]:
```
[(1,
  'Empresa A',
  '11.111.111/1111-11',
  'São Paulo',
  'SP',
  '11111-111',
  '2020-01-01'),
 (2,
  'Empresa B',
  '22.222.222/2222-22',
  'Rio de Janeiro',
  'RJ',
  '22222-222',
  '2020-01-01')]
```

In [ ]:
```
cursor.execute("SELECT * FROM fornecedor WHERE id_fornecedor = 5")
resultado = cursor.fetchall()
print(resultado)

cursor.close()
conn.close()
```

```
[]
```

UPDATE

Ao inserir um registro no banco, pode ser necessário alterar o valor de uma coluna, o que pode ser feito por meio da instrução SQL UPDATE. Observe o código a seguir.

In [ ]:
```
import sqlite3

conn = sqlite3.connect('aulaDB.db')
cursor = conn.cursor()

cursor.execute("UPDATE fornecedor SET cidade = 'Campinas' WHERE id_fornecedor =
conn.commit()

cursor.execute("SELECT * FROM fornecedor")
for linha in cursor.fetchall():
    print(linha)

cursor.close()
conn.close()
```

```
(1, 'Empresa A', '11.111.111/1111-11', 'São Paulo', 'SP', '11111-111', '2020-01
-01')
(2, 'Empresa B', '22.222.222/2222-22', 'Rio de Janeiro', 'RJ', '22222-222', '20
20-01-01')
(3, 'Empresa C', '33.333.333/3333-33', 'Curitiba', 'PR', '33333-333', '2020-01-
01')
```

DELETE

Ao inserir um registro no banco, pode ser necessário removê-lo no futuro, o que pode ser feito por meio da instrução SQL DELETE. Observe o código a seguir.

In [ ]:
```
import sqlite3

conn = sqlite3.connect('aulaDB.db')
cursor = conn.cursor()

cursor.execute("DELETE FROM fornecedor WHERE id_fornecedor = 2")
conn.commit()

cursor.execute("SELECT * FROM fornecedor")
for linha in cursor.fetchall():
    print(linha)

cursor.close()
conn.close()
```

```
(1, 'Empresa A', '11.111.111/1111-11', 'São Paulo', 'SP', '11111-111', '2020-01
-01')
(3, 'Empresa C', '33.333.333/3333-33', 'Curitiba', 'PR', '33333-333', '2020-01-
01')
```

INFORMAÇÕES DO BANCO DE DADOS E DAS TABELAS

Além das operações de CRUD, é importante sabermos extrair informações estruturais do banco de dados e das tabelas. Por exemplo, considerado um banco de dados, quais tabelas existem ali? Quais são os campos de uma tabela? Qual é a estrutura da tabela, ou seja, qual DDL foi usada para gerá-la? Os comandos necessários para extrair essas informações podem mudar entre os bancos, mas vamos ver como extraí-las do SQLite.

No código a seguir (entrada 11), temos uma instrução SQL capaz de retornar as tabelas no banco SQLite (linha 8) e outra capaz de extrair as DDLs usadas para gerar as tabelas (linha 15).

In [ ]:
```
import sqlite3

conn = sqlite3.connect('aulaDB.db')
cursor = conn.cursor()

# Lista as tabelas do banco de dados
cursor.execute("""SELECT name FROM sqlite_master WHERE type='table' ORDER BY nam
print('Tabelas:')
for tabela in cursor.fetchall():
    print(tabela)

# Captura a DDL usada para criar a tabela
tabela = 'fornecedor'
cursor.execute(f"""SELECT sql FROM sqlite_master WHERE type='table' AND name='{t
print(f'\nDDL da tabela {tabela}:')
for schema in cursor.fetchall():
    print("%s" % (schema))

cursor.close()
conn.close()
```

```
Tabelas:
('fornecedor',)
('sqlite_sequence',)

DDL da tabela fornecedor:
CREATE TABLE fornecedor (
    id_fornecedor INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    nome_fornecedor TEXT NOT NULL,
    cnpj VARCHAR(18) NOT NULL,
    cidade TEXT,
    estado VARCHAR(2) NOT NULL,
    cep VARCHAR(9) NOT NULL,
    data_cadastro DATE NOT NULL
)
```

DESAFIO

Como desenvolvedor em uma empresa de consultoria de software, você foi alocado para construir uma solução parametrizável capaz de criar banco de dados na tecnologia SQLite, criar e apagar tabelas, o nome do banco, das tabelas e a DDL necessária para criar uma tabela (tudo deve ser parametrizável). Você também deve construir uma solução capaz de fazer um CRUD em um banco SQLite – veja que o componente deve funcionar para qualquer nome de banco e de tabela, ou seja, parâmetros. As regras que lhe foram passadas são as seguintes:

- Para criar uma nova base de dados, é necessário informar o nome.
- Para criar uma nova tabela, é necessário informar o nome do banco que receberá a tabela e a DDL de criação.
- Para excluir uma tabela, é necessário informar o nome do banco e da tabela.

- Para inserir um novo registro em uma tabela, é preciso informar: o nome do banco e da tabela e um dicionário contendo a chave e o valor a ser inserido. Por exemplo, {'nome': 'João', 'idade': 30}.
- Para recuperar, os dados é preciso informar o nome do banco e da tabela.
- Para atualizar um novo registro em uma tabela é preciso informar: o nome do banco e da tabela e dois dicionários, um contendo a chave e o valor a ser atualizado e outro contendo a chave e o valor da condição do registro que será atualizado.
- Para excluir um registro em uma tabela, é preciso informar: o nome do banco e da tabela e um dicionário contendo a chave e o valor da condição para localizar o registro a ser inserido. Por exemplo, {'id_cliente': 10}.